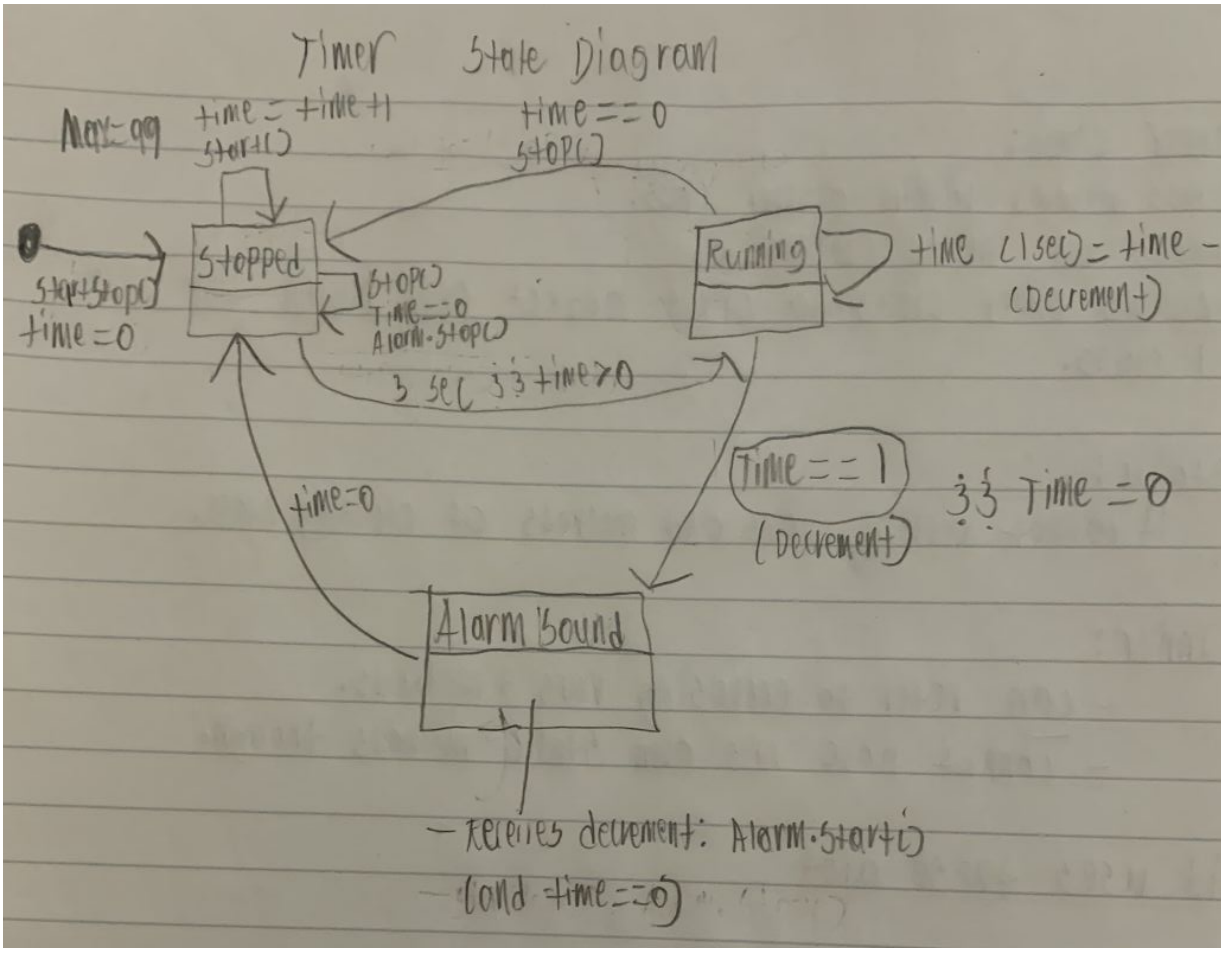
Written Part/Documentation (20%)

Please include these deliverables in the **doc** folder of your project 4.

- (0.5) Include the model from a future in-class group activity. (Minimally, a cell phone scan of your drawing is acceptable.)



- (0.25) Use inline comments to document design details in the code.
- (0.25) Use javadoc comments to document how to use the abstractions (interfaces, classes) you designed.

- Include a brief (300-500 words) report (0.5) on your pair development journey during this project. Focus on aspects you find noteworthy, e.g., process, pairing, testing, design decisions, refactoring, use of the repository.

For Project 4, our task was to create a simple countdown timer that had an 88-display and a multifunction button. The multifunction button mainly has three functions that occur in two different states. Initially, the timer is in its Stopped State where the time is at zero seconds. In the Stopped State, the user has the ability to increment the time by one second, essentially an add-time function. After the user has specified the amount of time that they added and after three seconds of inactivity, the timer then begins counting down and enters its Running State.
While the timer is in the Running State, the button's function becomes a cancel button, resetting the timer back to zero seconds if it is pressed; otherwise the timer will continue to countdown until it reaches zero seconds thus triggering the alarm. The alarm will continue to sound unless the button, whose function is now a stop/reset button is pressed.

Through the use of video and text chat via the Discord communications platform as well as the combination of Bitbucket as our Git repository and Android Studio as our IDE, we as a group were able to collaborate efficiently despite not being able to see each other. The skeleton code for this project was originally not what we wanted to start with so we modified the code to make it more simple, especially for the display.
After getting everything setup, we first needed to create a way to have the timer increase one second by tapping / clicking the multi-function button when running. This was done by creating a class to increment the time in the state section. Our next challenge was to make the timer start at zero and have the timer count down (decrement) instead of up (increment). The timer class that is imported for the project was especially useful for this part. This was done by creating a decrementing state class in the state section. Our final challenge for this project which was likely the most difficult part of the project was to have the timer actually start to countdown after three seconds of not clicking / tapping the application when running. This was done by furthering the modification of the incrementing state to catch the run time and determining if it has passed three seconds or not.

(0.5) The relationship between your state machine model from this project and your actual code. Possible talking points are:

- What are the similarities and differences between model and code?
    - The similarities between the model and the code include how the process should flow. For example, the model shows how at the start of the program the user has the option to continuously click on the multifunction button to add time by one second. The code also shows that at the start of the program, the ability to increment the time by one second is given as long as the time is equal to or below 99 seconds. Additionally, the model

shows the different states the program goes through as it runs. Similarly, the actual code is divided into classes which represent the different states of the program. In regard to the differences between the model and the code, the code is obviously more verbose and detailed than the model. Additionally, the model gives you the “big picture” of the program whereas the code is more concerned with the functionality of the program.

- Is it more effective to code or model first?
    - Ideally, it is more effective to model first such that it provides the user with a visualization of the processes/states that the program is running through. In effect, by modeling first you are essentially creating an outline of the program from its initialization to its termination.
- Now that you have the code, what, if any, changes would you make to your model?
    - Looking at the model right now, we can clearly see that we are missing 2 different states that we added in. These states would be incrementing and decrementing. We found it beneficial to include these states in our code, since we figured that transitioning to these 2 classes would help make our code operate more smoothly. Having separate states for incrementing the amount of time present on the screen and for slowly decrementing the time makes for a more efficient application. Looking back at when we were designing the model, we didn’t really think about having these 2 states be separate entities on the diagram we drew up.